Puesta en producción de modelos de aprendizaje automático

Presentación con una selección de las slides de las dos primeras presentaciones de Chip Huyen que se pueden consultar en stanford-cs329s.github.io/syllabus.html

How to pass data between processes?

Ride management

Need driver availability & price to show riders

Driver management

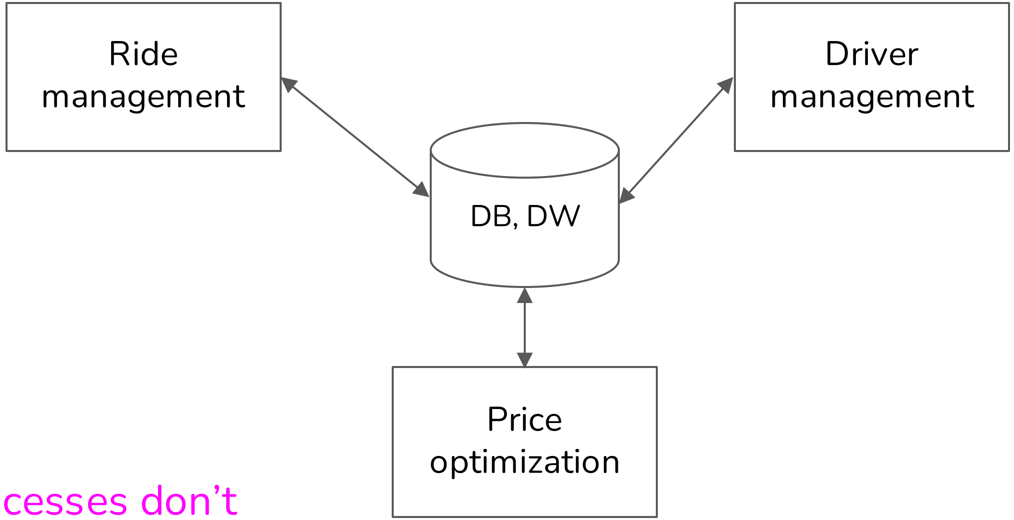
Need ride demand & price to incentivize drivers

Price optimization

Need ride demand & driver availability to set price

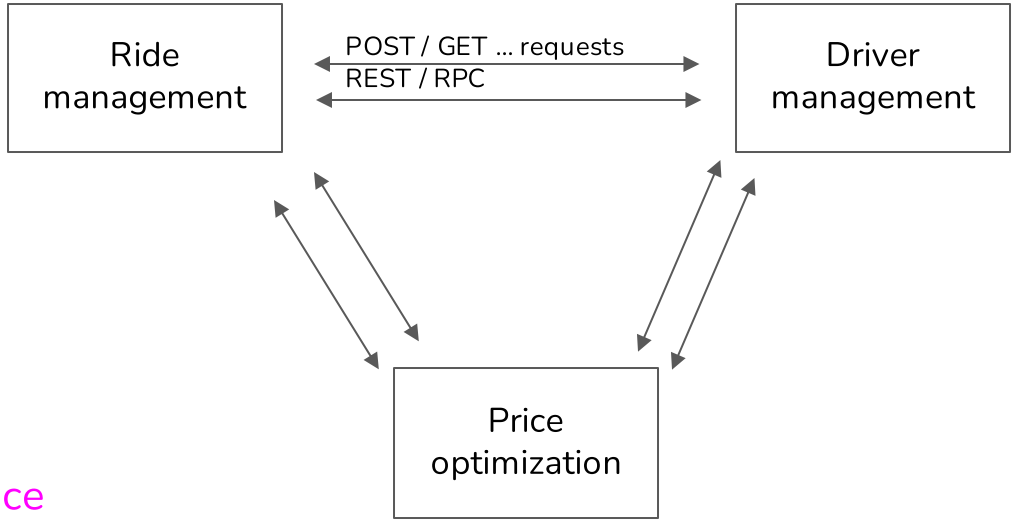
A simple ride-sharing microservice

Data passing through databases



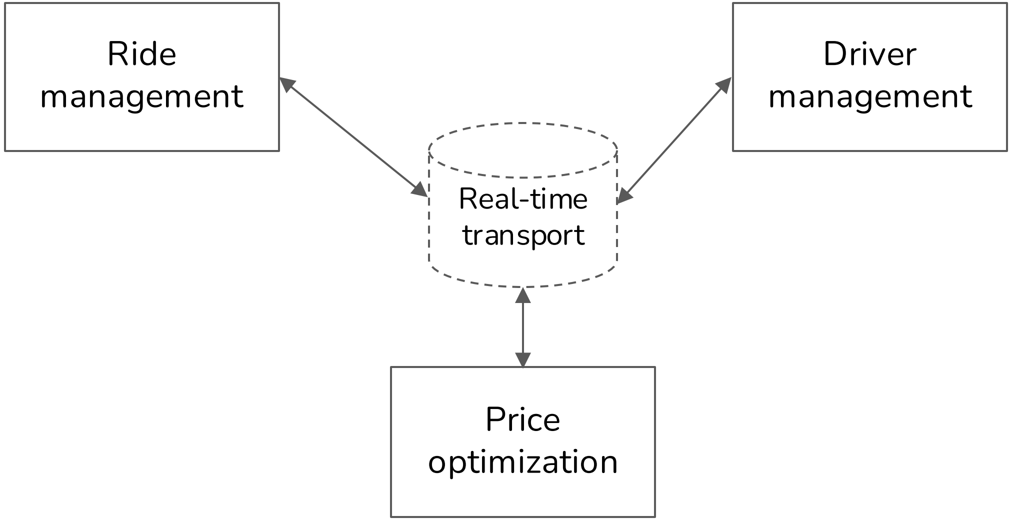
1. What if processes don't share database access?
2. Read & write from databases can be slow

Data passing through services



Inter-service communication can blow up

Data passing through brokers



Real-time transport: pubsub

- Any service can publish to a stream [producer]
- Any service can subscribe to a stream to get info they need [consumer]

```
from confluent_kafka import Consumer, SerializingProducer

producer = SerializingProducer(producer_config)
producer.produce(
            topic="prediction",
            key=example_id,
            value=prediction,
         )

pred_consumer = Consumer(consumer_config)
pred_consumer.subscribe(["prediction"])
```

Real-time transport: pubsub, message queue, etc.

APACHE kafka®

1240 companies reportedly use **Kafka** in their tech stacks, including **Uber**, **Shopify**, and **Spotify**.

Uber | Shopify | Spotify | Udemy | Robinhood | Slack | LaunchDarkly | Nubank | The New York Times

RabbitMQ

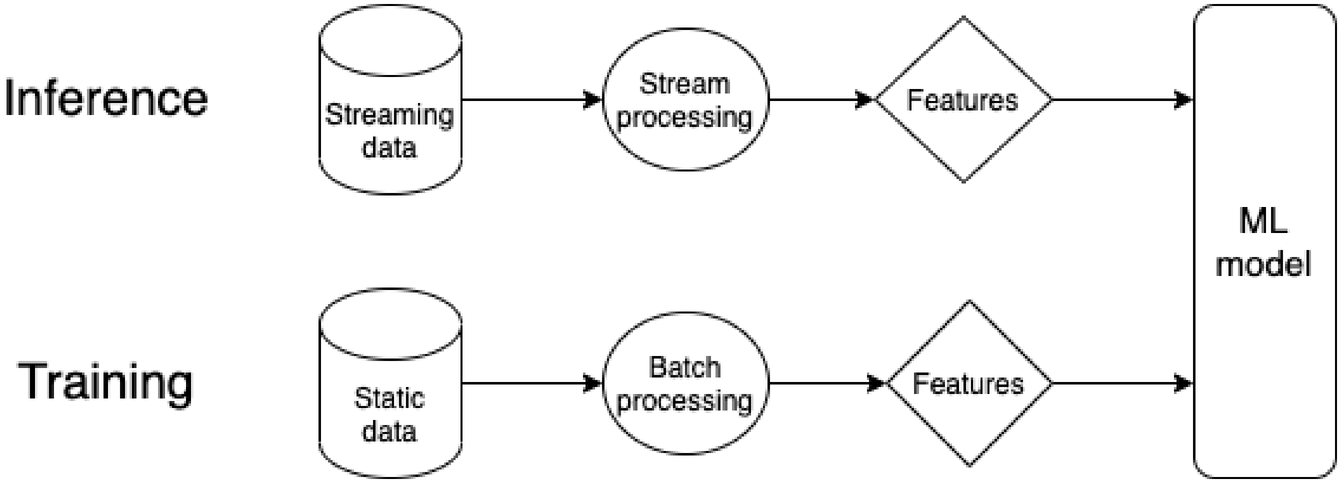
1811 companies reportedly use **RabbitMQ** in their tech stacks, including **Robinhood**, **reddit**, and **Stack**.

Robinhood | reddit | Stack | Accenture | Hepsiburada | CircleCI | Alibaba Travels | trivago | ViaVarejo

Batch processing vs. stream processing

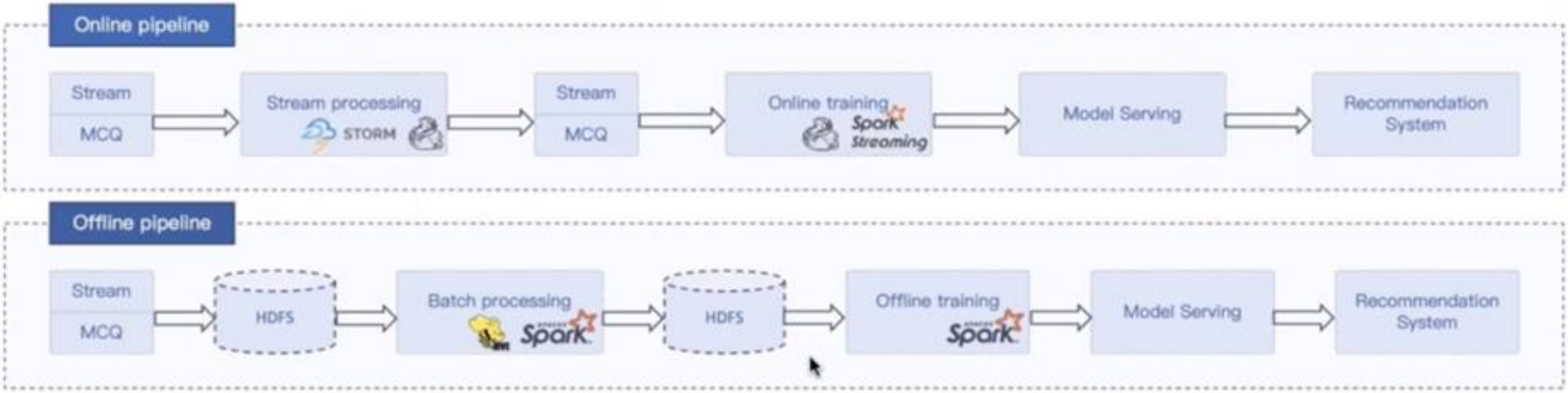
Historical data	Streaming data
Databases, data warehouses	Kafka, Kinesis, Pulsar, etc.
Batch features: ● age, gender, job, city, income ● when account was created	Dynamic features ● locations in the last 10 minutes ● recent activities
Bounded: know when a job finishes	Unbounded: never finish
Processing kicked of periodically, in batch ● e.g. MapReduce, Spark	Processing can be kicked off as events arrive ● e.g. Flink, Samza, Spark Streaming

One model, two pipelines



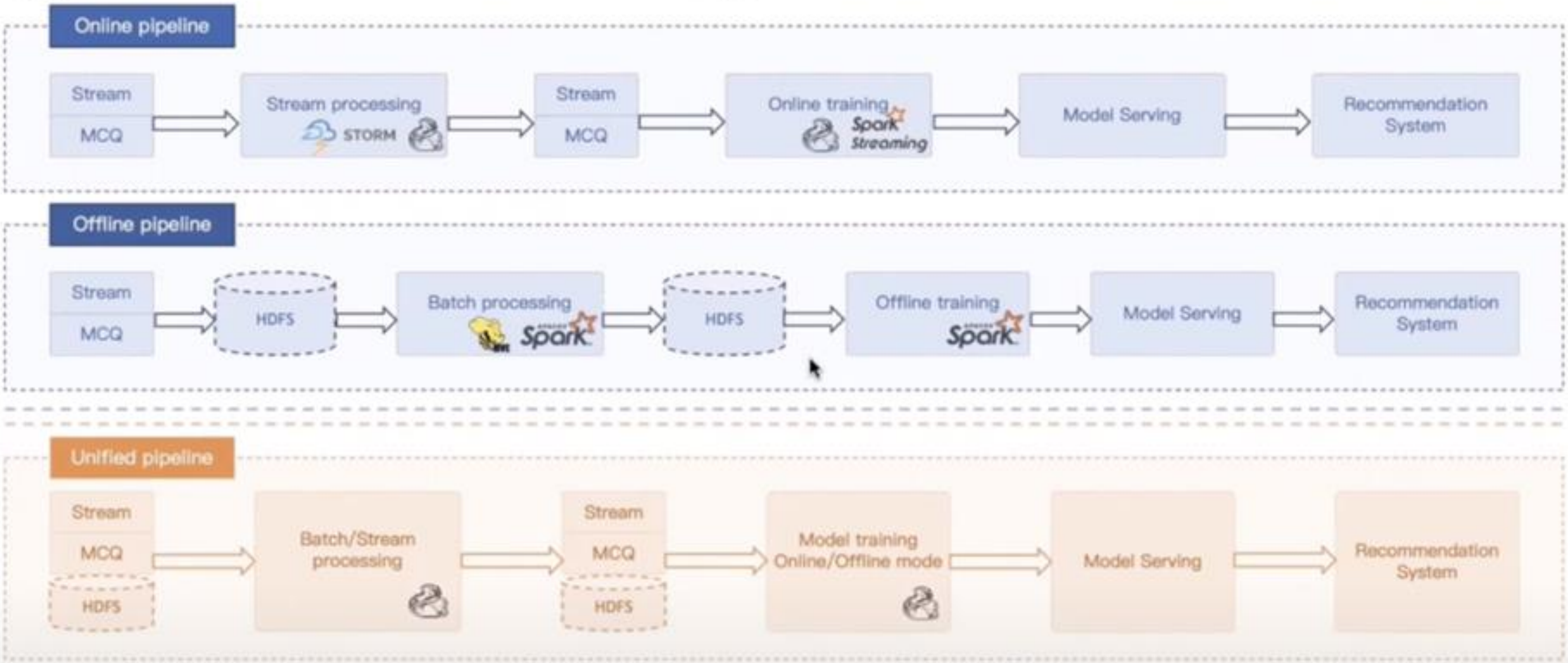
⚠⚠ A common source of errors in production ⚠⚠

One model, two pipelines: example



Machine learning with Flink in Weibo (Qian Yu, QCon 2019)

Apply unified Flink APIs to both online and offline ML pipelines



Machine learning with Flink in Weibo (Qian Yu, QCon 2019)

Barriers to stream processing

1. Companies don’t see the benefits of streaming
    - Systems not at scale
    - Batch predictions work fine
    - Online predictions would work better but they don’t know that

Barriers to stream processing

1. Companies don’t see the benefits of streaming
2. High initial investment on infrastructure
3. Mental shift
4. Python incompatibility

How to serve your model?

1. Batch prediction vs. online prediction
2. Cloud computing vs. edge computing

⚠️⚠️ The dangers of categorical thinking ⚠️⚠️

- Seemingly different ways of doing things might be fundamentally similar
- Choices don't have to be mutually exclusive
- Choices can evolve over time

The Dangers of Categorical Thinking

We're hardwired to sort information into buckets—and that can hamper our ability to make good decisions. by Bart de Langhe and Philip Fernbach

Batch prediction vs. online prediction

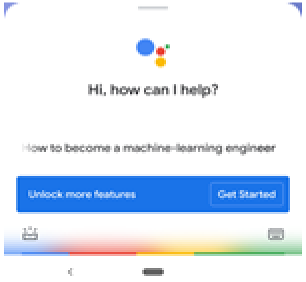
Batch prediction vs. online prediction

- Batch prediction
  - Generate predictions periodically before requests arrive
  - Predictions are stored (e.g. SQL tables) and retrieved when requests arrive
  - Asynch
- Online prediction
  - Generate predictions after requests arrive
  - Predictions are returned as responses
  - Sync when using requests like REST / RPC
    - HTTP prediction
  - Async (with low latency) with real-time transports like Kafka / Kinesis
    - Streaming prediction

Offered by major cloud providers

Still challenging

	Batch prediction (async)	Online prediction (generally sync)
Frequency	Periodical (e.g. every 4 hours)	As soon as requests come
Useful for	Processing accumulated data when you don't need immediate results (e.g. recommendation systems)	When predictions are needed as soon as data sample is generated (e.g. fraud detection)
Optimized	High throughput	Low latency
Input space	Finite: need to know how many predictions to generate	Can be infinite
Examples	● TripAdvisor hotel ranking ● Netflix recommendations	● Google Assistant speech recognition ● Twitter feed



Hybrid: batch & online prediction

- Online prediction is default, but common queries are precomputed and stored

  DOORDASH
    - Restaurant recommendations use batch predictions
    - Within each restaurant, item recommendations use online predictions

  NETFLIX
    - Title recommendations use batch predictions
    - Row orders use online predictions

Cloud computing vs. edge computing

	Cloud computing	Edge computing
Computations	Done on cloud (servers)	Done on edge devices (browsers, phones, tablets, laptops, smart watches, activity watchers, cars, etc.)
Examples	• Most queries to Alexa, Siri, Google Assistant • Google Translate for rare language pairs (e.g. English - Yiddish)	• Wake words for Alexa, Siri, Google Assistant • Google Translate for popular language pairs (e.g. English - Spanish) • Predictive text • Unlocking with fingerprints, faces

Benefits of edge computing

- Can work without (Internet) connections or with unreliable connections
  - Many companies have strict no-Internet policy
  - **Caveat**: devices are capable of doing computations but apps need external information
    - e.g. ETA needs external real-time traffic information to work well

Benefits of edge computing

- Can work without (Internet) connections or with unreliable connections
    - Many companies have strict no-Internet policy
    - **Caveat**: devices are capable of doing computations but apps need external information
        - e.g. ETA needs external real-time traffic information to work well
- Don't have to worry about network latency
    - Network latency might be a bigger problem than inference latency
    - Many use cases are impossible with network latency
        - e.g. predictive texting

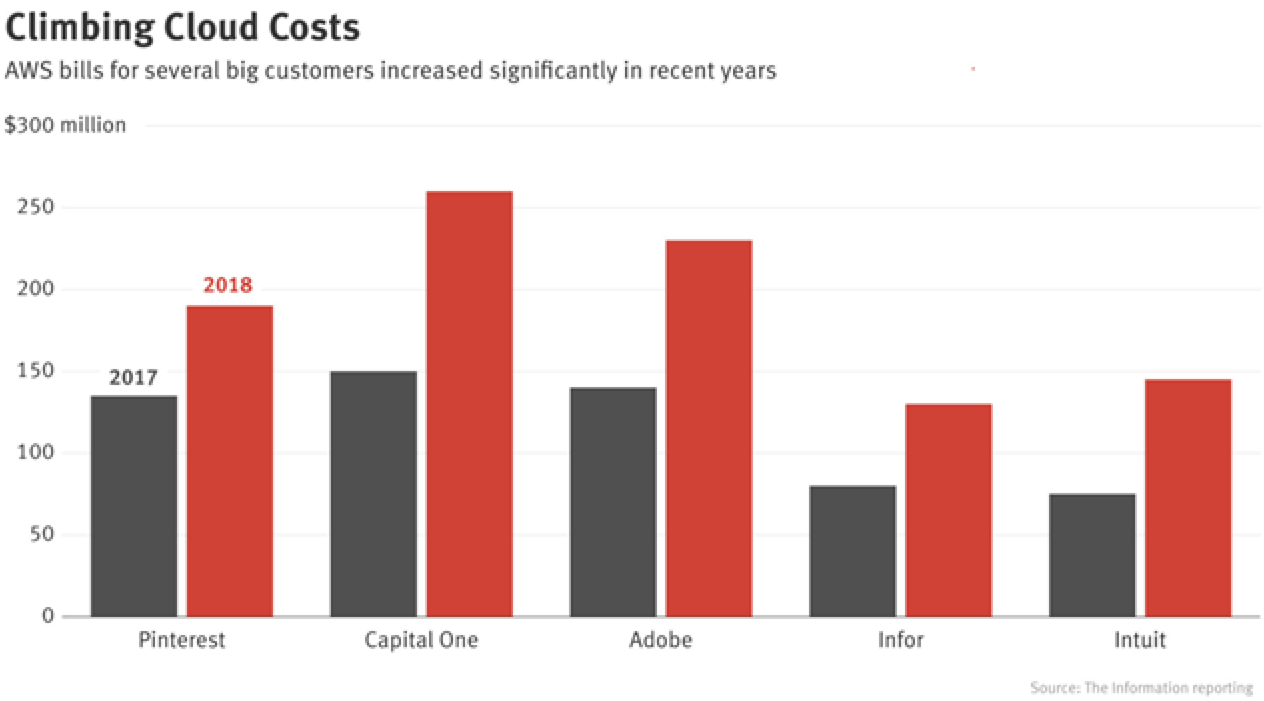
Benefits of edge computing

- Can work without (Internet) connections or with unreliable connections
  - Many companies have strict no-Internet policy
  - **Caveat**: devices are capable of doing computations but apps need external information
    - e.g. ETA needs external real-time traffic information to work well
- Don't have to worry about network latency
  - Network latency might be a bigger problem than inference latency
  - Many use cases are impossible with network latency
    - e.g. predictive texting
- Fewer concerns about privacy
  - Don't have to send user data over networks (which can be intercepted)
  - Cloud database breaches can affect many people
  - Easier to comply with regulations (e.g. GDPR)
  - **Caveat**: edge computing might make it easier to steal user data by just taking the device

Benefits of edge computing

- Can work without (Internet) connections or with unreliable connections
  - Many companies have strict no-Internet policy
  - **Caveat**: devices are capable of doing computations but apps need external information
    - e.g. ETA needs external real-time traffic information to work well
- Don't have to worry about network latency
  - Network latency might be a bigger problem than inference latency
  - Many use cases are impossible with network latency
    - e.g. predictive texting
- Fewer concerns about privacy
  - Don't have to send user data over networks (which can be intercepted)
  - Cloud database breaches can affect many people
  - Easier to comply with regulations (e.g. GDPR)
  - **Caveat**: edge computing might make it easier to steal user data by just taking the device
- Cheaper
  - The more computations we can push to the edge, the less we have to pay for servers

A cloud mistake can bankrupt your startup!



Hybrid

- Common predictions are <u>precomputed and stored</u> on device
- Local data centers: e.g. each warehouse has its own server rack
- Predictions are generated on cloud and <u>cached</u> on device

Challenges of ML on the edge

1. Hardware: Make hardware more powerful
2. Model compression: Make models smaller
3. Model optimization: Make models faster

Make hardware more powerful: big companies

Musk Boasts Tesla Has 'Best Chip in the World'

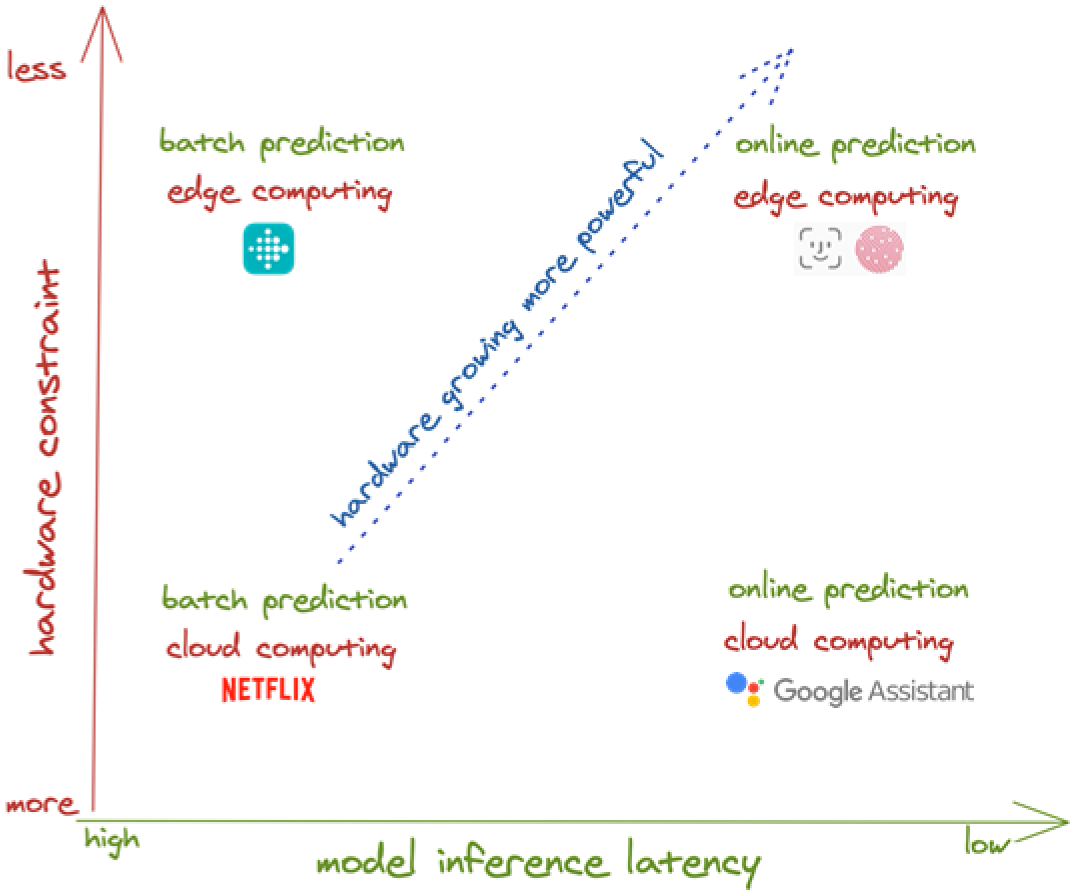
The CEO's newest big prediction: that Tesla will have self-driving cars on the road next year.

Bloomberg
APR 23, 2019

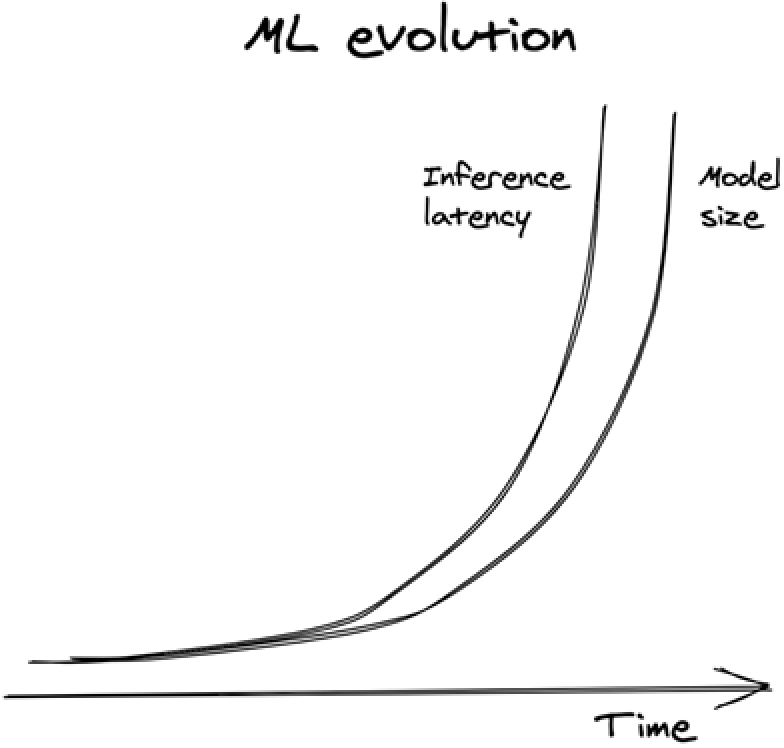
Apr 14, 2020 - Technology

Scoop: Google readies its own chip for future Pixels, Chromebooks

Future of ML: online and on-device



Model Compression



Bigger, better, slower

Model compression

1. Quantization
2. Knowledge distillation
3. Pruning
4. Low-ranked factorization

Model compression: active research/development

The Top 121 Model Compression Open Source Projects on Github

Categories > Machine Learning > **Model Compression**

Nni 10910 ⭐
An open source AutoML toolkit for automate machine learning lifecycle, including feature engineering, neural architecture search, model compression and hyper-parameter tuning.

Pocketflow 2676 ⭐
An Automatic Model Compression (AutoMC) framework for developing smaller and faster AI applications.

Neuronblocks 1404 ⭐
NLP DNN Toolkit - Building Your NLP DNN Models Like Playing Lego

Ghostnet 1823 ⭐
CV backbones including GhostNet, TinyNet and TNT, developed by Huawei Noah's Ark Lab.

Channel Pruning 1021 ⭐
Channel Pruning for Accelerating Very Deep Neural Networks (ICCV'17)

Model Optimization 1189 ⭐
A toolkit to optimize ML models for deployment for Keras and TensorFlow, including quantization and pruning.

Knowledge Distillation Pytorch 1291 ⭐
A PyTorch implementation for exploring deep and shallow knowledge distillation (KD) experiments with flexibility

Awesome Pruning 1361 ⭐
A curated list of neural network pruning resources.

Awesome Knowledge Distillation 1612 ⭐
Awesome Knowledge-Distillation. 分类整理的知识蒸馏paper(2014-2021)。

https://opensourcelibs.com/libs/model-compression | https://awesomeopensource.com/projects/model-compression

Model compression: quantization

- Reduces the size of a model by using fewer bits to represent parameter values
  - E.g. half-precision (16-bit) or integer (8-bit) instead of full-precision (32-bit)

100M params

32-bit
400MB

16-bit
200MB

Model compression: quantization

Pros	Cons

1. Reduce memory footprint
2. Increase computation speed
    a. Bigger batch size
    b. Computation on 16 bits is faster than on 32 bits
	1. Smaller range of values
	2. Values rounded to 0
	
	Need efficient rounding/scaling techniques

BFloat16: The secret to high performance on Cloud TPUs

Model compression: knowledge distillation

- Train a small model ("student") to mimic the results of a larger model ("teacher")
  - Teacher & student can be trained at the same time.
  - E.g. DistillBERT, reduces size of BERT by 40%, and increases inference speed by 60%, while retaining 97% language understanding.

DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter (Sanh et al., 2019)

Model compression: knowledge distillation

- Train a small model (“student”) to mimic the results of a larger model (“teacher”)
- Pros:
  - Fast to train student network if teacher is pre-trained.
  - Teacher and student can be completely different architectures.
- Cons:
  - If teacher is not pre-trained, may need more data & time to first train teacher.
  - Sensitive to applications and model architectures.

Model compression: pruning

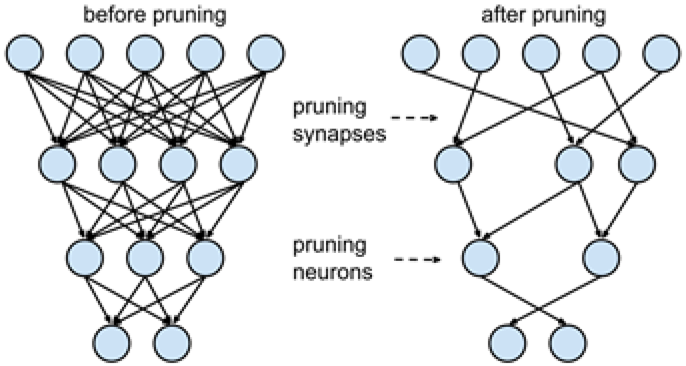
- Originally used for decision trees to remove uncritical sections
- Neural networks: reducing over-parameterization

Model compression: pruning methods

1. Remove nodes
    a. Changing architectures & reducing number of params

Model compression: pruning methods

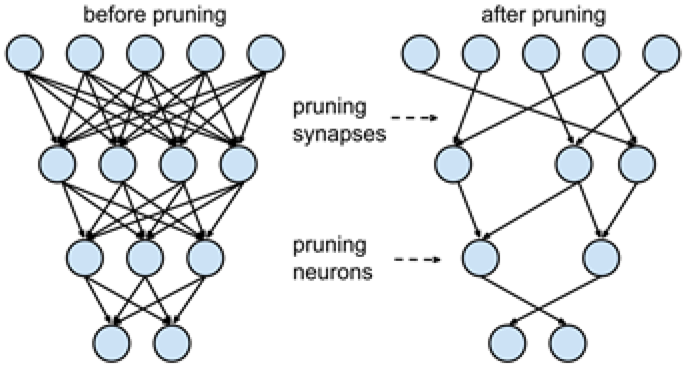
1. Remove nodes
2. Find least useful params & set to 0
   a. Number of params remains the same
   b. Reducing number of non-zero params



[Learning both Weights and Connections for Efficient Neural Networks](#) (Han et al., 2015)

Model compression: pruning methods

1. Remove nodes
2. Find least useful params & set to 0 ???
   a. Number of params remains the same
   b. Reducing number of non-zero params



Learning both Weights and Connections for Efficient Neural Networks (Han et al., 2015)

Model compression: pruning methods

1. Remove nodes
2. Find least useful params & set to 0
    a. Number of params remains the same
    b. Reducing number of non-zero params

Makes models more sparse

- lower memory footprint
- increased inference speed

The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks (Frankle et al., ICLR 2019)

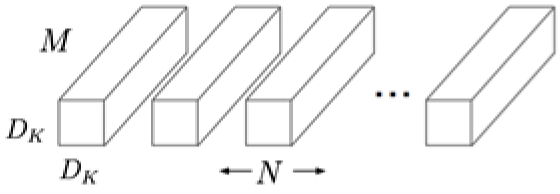
Model compression: pruning methods

1. Remove nodes
2. Find least useful params & set to 0
    a. Number of params remains the same
    b. Reducing number of non-zero params

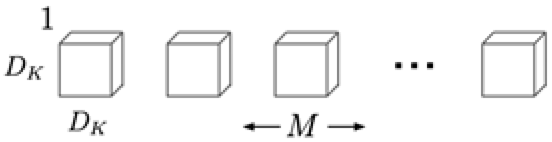
Can be used for architecture search

Model compression: factorization

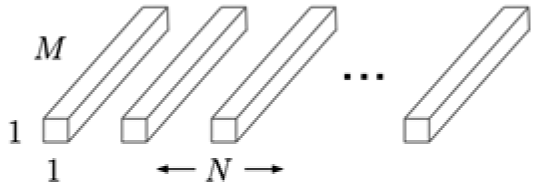
- 3 x 3 matrix can be written as a product of 3 x 1 and 1 x 3
  - 6 params instead of 9
- Replace convolution filters (many parameters) with compact blocks
  - E.g. MobileNets:
    - (a) are replaced by depthwise convolution
    - (b) and pointwise convolution
    - (c) to build a depthwise separable filter



(a) Standard Convolution Filters



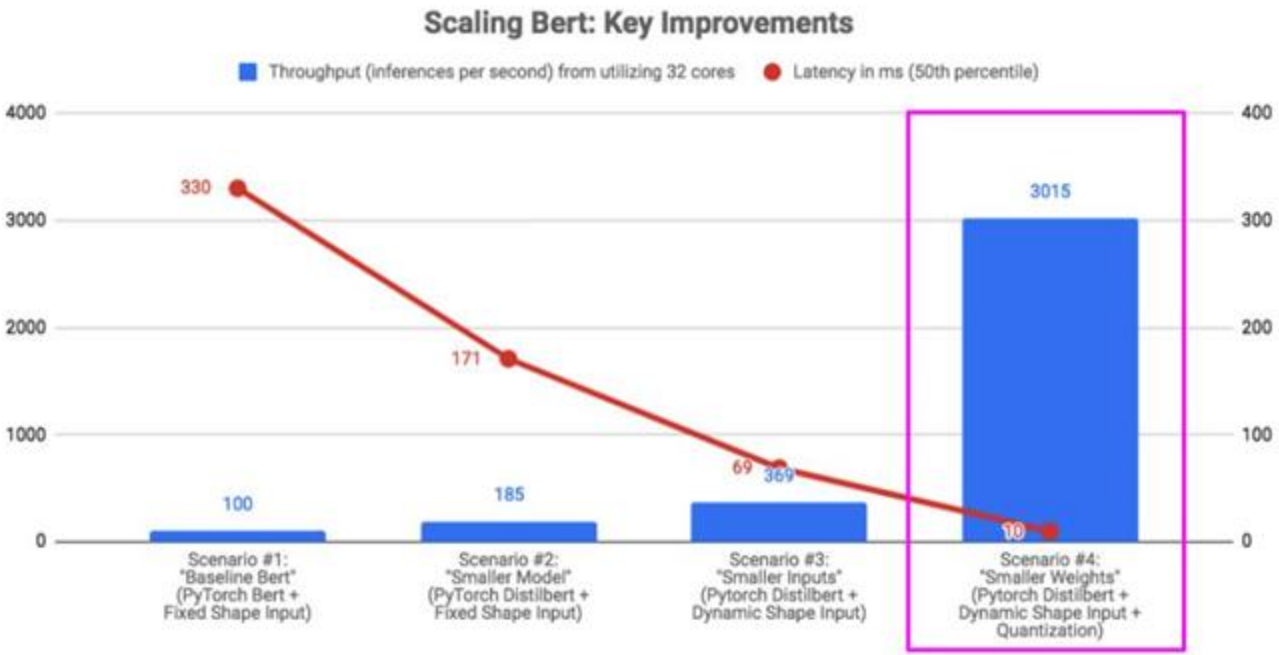
(b) Depthwise Convolutional Filters



(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications (Howard et al., 2017)

Make models smaller: case study



How We Scaled Bert To Serve 1+ Billion Daily Requests on CPUs (Roblox, 2020)

Compiling & optimizing models for edge devices

"With PyTorch and TensorFlow, you've seen the frameworks sort of converge. The reason quantization comes up, and a bunch of other lower-level efficiencies come up, is because $\boxed{\text{the next war is compilers for the frameworks}}$ — XLA, TVM, PyTorch has Glow, a lot of innovation is waiting to happen," he said. "For the next few years, you're going to see … how to quantize smarter, how to fuse better, how to use GPUs more efficiently, [and] how to automatically compile for new hardware."

Soumith Chintala, creator of PyTorch (VentureBeat, 2020)